NPSCS-94-005

NAVAL POSTGRADUATE SCHOOL
Monterey, California

AD-A278 653

A Lower Bound for the Intersection of Regular Forests

by Dennis M. Volpano

October 1993

94-12699

Approved for public release; distribution is unlimited.

Prepared for:

Naval Postgraduate School
Monterey. California 93943

94 4 25 111

Best
Available
Copy

NAVAL POSTGRADUATE SCHOOL

Monterey, California

REAR ADMIRAL T. A. MERCER
Superintendent

HARRISON SHULL
Provost

This report was prepared with research funded by the Naval Research Laboratory under the Reimbursable Funding.

This report was prepared by:

Dennis M. Volpano
Professor of Computer Science

Reviewed by:

Yutaka Kanayama
Associate Chairman for
Technical Research

Released by:

PAUL MARTO
Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPSCS-94-005	5. MONITORING ORGANIZATION REPORT NUMBER(S) Naval Postgraduate School
6a. NAME OF PERFORMING ORGANIZATION Computer Science Dept. Naval Postgraduate School 6b. OFFICE SYMBOL (if applicable) CS	7a. NAME OF MONITORING ORGANIZATION Naval Research Laboratory
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Postgraduate School 8b. OFFICE SYMBOL (if applicable) NPS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER O&MN Direct Funding
8c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A Lower Bound for the Intersection of Regular Forests

12. PERSONAL AUTHOR(S)
Dennis M. Volpano

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 10/92 TO 9/93	October 1993 9	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	tree automata, computational complexity

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Regular ΣX-forests continue to play an important role in programming languages, specifically in the design of type systems. They arise naturally as terms of constructor-based, recursive data types in logic and functional languages. Deciding whether the intersection of a sequence of regular ΣX-forests is nonempty is an important problem in type inference. We show that this problem is PSPACE-hard and as a corollary that the problem of constructing a regular ΣX-grammar representing their intersection is PSPACE-hard.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dennis M. Volpano	22b. TELEPHONE (Include Area Code) (408) 656-3091 22c. OFFICE SYMBOL CSVo

A Lower Bound for the Intersection of Regular Forests

Dennis M. Volpano

Department of Computer Science
Naval Postgraduate School
Monterey, California, USA

email: volpano@cs.nps.navy.mil

October 5, 1993

Abstract

Regular ΣX-forests continue to play an important role in programming languages, specifically in the design of type systems [MiR85, AM91, Vol93]. They arise naturally as terms of constructor-based, recursive data types in logic and functional languages. Deciding whether the intersection of a sequence of regular ΣX-forests is nonempty is an important problem in type inference. We show that this problem is PSPACE-hard and as a corollary that the problem of constructing a regular ΣX-grammar representing their intersection is PSPACE-hard.

1 Introduction

Regular ΣX-forests are playing an increasingly important role in language design and in particular in the design of type systems. Type inference then usually relies upon various operations over regular forests, one of which is *RF-INT* , deciding the emptiness of their intersection.

Definition 1.1 *The problem RF-INT is given a sequence of regular ΣX-grammars $G_1, \ldots, G_m$, decide whether $\bigcap_{k=1}^{m} T(G_k)$ is nonempty.*

Regular forests have been used to characterize the types of logic and functional programs [Mis84, MiR85, HeJ90, AM91] as well as overloadings introduced through classes in *Haskell* [Kae88, Vol93]. For example, Heintze and Jaffar propose what amounts to regular ΣX-grammars as inferred "types" or approximations of the semantics of logic programs. Corresponding to a logic program, say

$$\begin{aligned}&p(a).\\&p(f(X)) \leftarrow p(X).\\&r(b).\\&r(f(Y)) \leftarrow r(Y).\\&q(Z) \leftarrow p(Z), r(Z).\end{aligned}$$

is a set of equations

$$\begin{aligned}X &= a \cup f(X)\\Y &= b \cup f(Y)\\Z &= X \cap Y\end{aligned}$$

whose simultaneous least fixed point is an approximate meaning of the program. The inferred approximation or "type" is given by

$$\begin{aligned}X &= a \cup f(X)\\Y &= b \cup f(Y)\\Z &= \emptyset\end{aligned}$$

Solving for variable Z requires deciding whether the intersection of the two regular forests described by the first two equations is nonempty.

One can also view the logic program above as describing a set of valid overloadings in *Haskell* for p and r as operators where p has instances at types a and f, and r at b and f:

$$\begin{aligned}&\mathbf{class}\ P\,\alpha\ \mathbf{where}\ p :: \alpha\\&\mathbf{instance}\ P\,a\ \mathbf{where}\ p = \ldots\\&\mathbf{instance}\ P\,X\ \Rightarrow\ P\,f(X)\ \mathbf{where}\ p = \ldots\end{aligned}$$

$$\begin{aligned}&\mathbf{class}\ R\,\alpha\ \mathbf{where}\ r :: \alpha\\&\mathbf{instance}\ R\,b\ \mathbf{where}\ r = \ldots\\&\mathbf{instance}\ R\,Y\ \Rightarrow\ R\,f(Y)\ \mathbf{where}\ r = \ldots\end{aligned}$$

Instance declarations for an overloaded operator in *Haskell* describe a regular forest. So for example, deciding whether term $p = r$ is typable requires

deciding whether the regular forest arising from p's instance declarations intersects with the forest described by instances for r.

2 Forests and Regular ΣX-grammars

Given an alphabet A, an A-valued tree t is specified by its set of nodes (the "domain" $dom(t)$) and a valuation of the nodes in A. Formally, a *k-ary*, A-valued tree is a map $t : dom(t) \to A$ where $dom(t) \subseteq \{0, \ldots, k-1\}^*$ is a nonempty set, closed under prefixes. The frontier of t is the set

$$\{w \in dom(t) \mid \neg\exists i. wi \in dom(t)\}.$$

It is assumed that A is partitioned into a *ranked alphabet* Σ and a *frontier alphabet* X. A ranked alphabet, or *signature*, is a finite nonempty operator domain. For any Σ and X, we denote the set of all *finite* ΣX-trees by $F_\Sigma(X)$. A forest, or tree language, $T \subseteq F_\Sigma(X)$ is called *regular* if and only if for some finite set C disjoint from Σ and X, T can be obtained from finite subsets of $F_\Sigma(X \cup C)$ by applications of union, concatenation $\cdot_c$ (defined using tree substitution), and closure ^{*c} where $c \in C$ [Tho90].

A regular forest can alternatively be defined as a tree language generated by a regular ΣX-grammar [GeS84].

Definition 2.1 *A regular ΣX-grammar G consists of*

- *a finite nonempty set N of nonterminal symbols,*
- *a finite set P of productions of the form $A \to r$ where $A \in N$ and $r \in F_\Sigma(N \cup X)$, and*
- *an initial symbol $S \in N$.*

Definition 2.2 *If $G = (N, \Sigma, X, P, S)$ is a regular ΣX-grammar then the ΣX-forest generated by G is*

$$T(G) = \{t \in F_\Sigma(X) \mid S \Rightarrow^*_G t\}$$

Regular ΣX-grammars are a class of context-free grammars that define the same family of forests as those recognized by nondeterministic root-to-frontier (NDR) ΣX-automata. A root-to-frontier automaton can be viewed

as an attribute evaluator for a tree whose attributes are states prescribed by an attribute grammar with *inherited* attributes only. Formally, a NDR ΣX-automaton $\mathbf{A}$ is a tuple $(\mathcal{A}, A', \alpha)$ such that

1. $\mathcal{A}$ is a finite NDR Σ-algebra (A, Σ),
2. $A' \subseteq A$ is a set of *initial states*, and
3. $\alpha : X \to \wp A$ is a *final assignment*.

In a NDR Σ-algebra (A, Σ), A is a nonempty set of states and every $\sigma \in \Sigma_m$ with $m \geq 1$ is realized as a mapping $\sigma^{\mathcal{A}} : A \to \wp(A^m)$. For $\sigma \in \Sigma_0$, $\sigma^{\mathcal{A}}$ is a subset of A.

For example, a NDR ΣX-automaton $\mathbf{A} = (\mathcal{A}, A', \alpha)$ recognizing set

$$\{\sigma(x,y),\ \sigma(y,x)\}$$

can be defined as follows. Let $\Sigma = \Sigma_2 = \{\sigma\}$, $X = \{x, y\}$, and the set of initial states $A' = \{S\}$. Define $\mathcal{A} = (\{\hat{x}, \hat{y}, S\}, \Sigma)$ such that

$$\sigma^{\mathcal{A}}(S) = \{(\hat{x}, \hat{y}), (\hat{y}, \hat{x})\}$$

and finally define the final assignment α as

$$\begin{aligned} x\alpha &= \{\hat{y}\} \\ y\alpha &= \{\hat{x}\} \end{aligned}$$

It is interesting to note that there is no deterministic root-to-frontier ΣX-automaton that accepts the set above. Suppose automaton $\mathbf{A}$ accepts $\sigma(x, y)$ and $\sigma(y, x)$ and that $\sigma(a) = (a_1, a_2)$ for some states a, a_1, and a_2 of $\mathbf{A}$. If α is $\mathbf{A}$'s final assignment function, then

$$x\alpha = a_1,\ \ y\alpha = a_2,\ \ y\alpha = a_1,\ \ x\alpha = a_2$$

Since $\mathbf{A}$ is deterministic, $a_1 = a_2$. So we have $\sigma(a) = (a_1, a_1)$ where $x\alpha = y\alpha = a_1$. Therefore on $\sigma(x, x)$ and $\sigma(y, y)$, $\mathbf{A}$ enters the leaves in state a_1 such that $a_1 \in x\alpha$, and $a_1 \in y\alpha$. Thus $\mathbf{A}$ accepts $\sigma(x, x)$ and $\sigma(y, y)$ as well.

Given that regular ΣX-grammars define exactly the forests recognized by NDR ΣX-automata, one could formulate *RF-INT* in terms of the latter representation of regular forests. But we choose regular ΣX-grammars instead since they are better suited for manipulation.

Regular forests are effectively closed under intersection.

Theorem 2.1 *If G_1 and G_2 are regular ΣX-grammars, for a given Σ and X, then $T(G_1) \cap T(G_2)$ is a forest generated by a regular ΣX-grammar.*

Proof. Suppose $G_1 = (N_1, \Sigma, X, P_1, S_1)$ and $G_2 = (N_2, \Sigma, X, P_2, S_2)$ are regular ΣX-grammars. Let ΣX-grammar $G = (N_1 \times N_2, \Sigma, X, P, [S_1, S_2])$ where

$$[A, B] \rightarrow a([Y_1, Z_1], \ldots, [Y_n, Z_n]) \in P, \quad \text{for} \quad n \geq 0$$

if and only if

$$A \rightarrow a(Y_1, \ldots, Y_n) \in P_1,$$
$$B \rightarrow a(Z_1, \ldots, Z_n) \in P_2,$$

and $a \in \Sigma$, or $[A, B] \rightarrow a \in P$ if and only if $a \in X$. Then $T(G) = T(G_1) \cap T(G_2)$. □

The theorem implies that the family of regular forests is properly contained within the context-free languages since the latter is not closed under intersection.

We now state and prove the main result.

Theorem 2.2 *RF-INT is PSPACE-hard.*

Proof. The proof uses a result of [Koz77]. For every deterministic Turing machine M of polynomial space complexity, we give a *log-space* transducer that on input x, outputs a sequence of regular ΣX-grammars whose intersection is nonempty iff M accepts x.

Let M be a single tape DTM of polynomial space complexity $p(n) \geq n$ and assume that M always makes at least three odd number of moves, has a unique accepting state, q_{acc}, and erases its tape before accepting, positioning its tape head at the left end of the tape. Let $x = a_1 \ldots a_n$ be a string over M's input alphabet and suppose M has states Q and tape symbols Γ such that Q, Γ, and set $\{nil, \#, \#\#\}$ are pairwise disjoint. If

$$\Delta = \Gamma \cup \{[qX] \mid q \in Q \;\&\; X \in \Gamma\}$$

then ranked alphabet $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$ where $\Sigma_0 = \{nil\}$, $\Sigma_1 = \Delta$, $\Sigma_2 = \{\#\#\}$ and $\Sigma_3 = \{\#\}$. Suppose ID_Δ derives regular forest

$$Z_1 \left(Z_2 \left(\cdots Z_{p(n)} \left(nil \right) \cdots \right) \right)$$

for all $Z_k \in \Delta$, $1 \leq k \leq p(n)$, and $ID_i^{[X_1X_2X_3]}$ derives regular forest

$$Z_1(\cdots Z_{i-1}(X_1(X_2(X_3(Z_i(\cdots Z_{p(n)-3}(nil)\cdots)$$

for all $X_1, X_2, X_3, Z_k \in \Delta$, $1 \leq k \leq p(n) - 3$.

A computation of M consists of a sequence of instantaneous descriptions $ID_0 \vdash ID_1 \vdash \cdots \vdash ID_{2m+1}$, each containing the contents of M's tape padded with blanks (B's) to length $p(n)$. If according to a move of M, symbols $Y_1Y_2Y_3$ in positions i, $i+1$, and $i+2$ respectively of an ID can follow from symbols $X_1X_2X_3$ in the same positions of another ID, we write

$$ID_i^{[X_1X_2X_3]} \vdash_M ID_i^{[Y_1Y_2Y_3]}$$

We give two regular ΣX-grammars F_i^{odd} and F_i^{even} such that F_i^{odd} ensures that even ID's follow from odd ones, and F_i^{even} that odd ones follow from even ones. Let F_i^{odd} be a regular ΣX-grammar with empty frontier alphabet, start symbol S and productions

$$S \rightarrow \#(ID_\Delta, ID_i^{[Z_1Z_2Z_3]}, F_i^{[Z_1Z_2Z_3]})$$

for all $Z_k \in \Delta$, $1 \leq k \leq 3$,

$$F_i^{[X_1X_2X_3]} \rightarrow \#(ID_i^{[Y_1Y_2Y_3]}, ID_i^{[Z_1Z_2Z_3]}, F_i^{[Z_1Z_2Z_3]})$$

for all $X_k, Y_k, Z_k \in \Delta$, $1 \leq k \leq 3$, such that $ID_i^{[X_1X_2X_3]} \vdash_M ID_i^{[Y_1Y_2Y_3]}$, and

$$F_i^{[X_1X_2X_3]} \rightarrow \#\#(ID_i^{[Y_1Y_2Y_3]}, ID_\Delta)$$

for all $X_k, Y_k \in \Delta$, $1 \leq k \leq 3$, such that $ID_i^{[X_1X_2X_3]} \vdash_M ID_i^{[Y_1Y_2Y_3]}$.

Let F_i^{even} be a regular ΣX-grammar with empty frontier alphabet, start symbol S and productions

$$S \rightarrow \#(ID_i^{[X_1X_2X_3]}, ID_i^{[Y_1Y_2Y_3]}, S)$$
$$S \rightarrow \#\#(ID_i^{[X_1X_2X_3]}, ID_i^{[Y_1Y_2Y_3]})$$

for all $X_k, Y_k \in \Delta$, $1 \leq k \leq 3$, such that $ID_i^{[X_1X_2X_3]} \vdash_M ID_i^{[Y_1Y_2Y_3]}$.

Finally, suppose *initID* derives the unary tree

$$[q_0a_1](a_2(\cdots a_n(B_{n+1}(\cdots B_{p(n)}(nil)\cdots)$$

where B_k is a blank and q_0 is the start state of M, and *finalID* derives

$$[q_{acc}B](B_2(\cdots B_{p(n)}(nil)\cdots)$$

Then let F_{end} be a regular grammar with start symbol S and productions

$$S \rightarrow \#(initID, ID_\Delta, F_{acc})$$
$$F_{acc} \rightarrow \#(ID_\Delta, ID_\Delta, F_{acc})$$
$$F_{acc} \rightarrow \#\#(ID_\Delta, finalID)$$

Then we have

$$u \in \bigcap_{i=1}^{p(n)-2} T(F_i^{odd})$$

iff $u = \#(ID_0, ID_1, \#(\cdots\#(ID_{2m-2}, ID_{2m-1}, \#\#(ID_{2m}, ID_{2m+1})\cdots)$ and from ID_{2k-1} follows ID_{2k} according to the transition rules of M for $1 \leq k \leq m$. Likewise,

$$u \in \bigcap_{i=1}^{p(n)-2} T(F_i^{even})$$

iff $u = \#(ID_0, ID_1, \#(\cdots\#(ID_{2m-2}, ID_{2m-1}, \#\#(ID_{2m}, ID_{2m+1})\cdots)$ and from ID_{2k} follows ID_{2k+1} according to the rules of M for $0 \leq k \leq m$. Then

$$T(F_{end}) \cap \bigcap_{i=1}^{p(n)-2} T(F_i^{odd}) \cap T(F_i^{even})$$

is nonempty iff M accepts x. □

As is the case for emptiness of intersection of a sequence of DFA's, the source for the hardness of *RF-INT* lies not in deciding emptiness but rather in computing the intersection of regular forests.

Corollary 2.3 *Given regular ΣX-grammars $G_1, \ldots, G_m$, constructing a regular ΣX-grammar G such that $T(G) = \bigcap_{k=1}^{m} T(G_k)$ is PSPACE-hard.*

Proof. The emptiness of $T(G)$ for a regular ΣX-grammar G is decidable in time $O(|G|^2)$ in the usual way. From the proof of Theorem 2.2 then every problem in PSPACE is P-time Turing reducible to the problem of constructing the intersection of a sequence of regular ΣX-grammars. □

A simple a'gorithm for constructing G is based on the usual construction of forming the cartesian product of reachable states as is suggested in the proof of Theorem 2.1 [AiM91]. It has worst-case time complexity exponential in m. Unfortunately this naive construction is likely the best we can do. It should be pointed out that for a fixed m, constructing G from $G_1, \ldots, G_m$ can be done in polynomial time.

Deciding whether some number of DFA's accept a common string can be done in nondeterministic linear space, but this does not appear to be true for *RF-INT*, which can be decided in deterministic exponential time. This suggests that a tighter lower bound exists for *RF-INT*.

References

[AM91] Aiken, A. and Murphy, B.: Static Type Inference in a Dynamically Typed Language, *Proc. 18th ACM Symposium on Principles of Programming Languages*, pp. 279–290, 1991.

[AiM91] Aiken, A. and Murphy, B.: Implementing Regular Tree Expressions, *Proc. 5th Conf. on Functional Programming Languages and Computer Architecture*, *LNCS* **523**, Springer-Verlag, pp. 427–447, 1991.

[GeS84] Gecseg, F. and Steinby M.: Tree Automata, Akademiai Kiado, Budapest Hungary, 1984.

[HeJ90] Heintze N. and Jaffar J.: A Finite Presentation Theorem for Approximating Logic Programs, *Proc. 17th ACM Symposium on Principles of Programming Languages*, pp. 197–209, 1990.

[Kae88] Kaes, S.: Parametric Overloading in Polymorphic Programming Languages, *Proc. 2nd European Symposium on Programming*, *LNCS* **300**, Springer-Verlag, pp. 131–144, 1988.

[Koz77] Kozen, D.: Lower Bounds for Natural Proof Systems, *Proc. 18th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, Long Beach, CA pp. 254–266, 1977.

[Mis84] Mishra, P.: Toward a Theory of Types in PROLOG, *Proc. 1st IEEE Symposium on Logic Programming*, pp. 289–298, 1984.

[MiR85] Mishra, P. and Reddy, U.: Declaration-free Type Checking, *Proc. 12th ACM Symposium on Principles of Programming Languages*, pp. 7–21, 1985.

[Tho90] Thomas,W.: Automata on Infinite Trees, Handbook of Theoretical Computer Science, Volume B, Formal Methods and Semantics, J. vanLeeuwen, Ed. pp. 165–184, 1990.

[Vol93] Volpano, D.: Haskell-style Overloading is NP-hard, *submitted for publication*, 1993. 5th IEEE Int'l Conf. Toulouse France

Distribution List

Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Library, Code 52 Naval Postgraduate School Monterey, CA 93943	2
Director of Research Administration Code 08 Naval Postgraduate School Monterey, CA 93943	1
Dr. Neil C. Rowe, Code CSRp Naval Postgraduate School Computer Science Department Monterey, CA 93943-5118	1
Prof. Robert B. McGhee, Code CSMz Naval Postgraduate School Computer Science Department Monterey, CA 93943-5118	1
Dr. Ralph Wachter Software Program Office of Naval Research 800 N. Quincy St. Arlington VA 22217-5000	2
Dr. Dennis Volpano, Code CSVo Naval Postgraduate School Computer Science Dept. Monterey, CA 93943-5118	20